A Very Easy Weekly Round

Milmon

Hailiang Junior High School

2024 年 8 月 17 日

全球知名算法竞赛

给定一个 CF 题目的题目编号，输出其在 CF 上的链接。

全球知名算法竞赛：子任务 $1 \sim 2$

从第三个字符开始扫描字符串，如果是字母就加入字符串 t_1，否则加入字符串 t_2。最后把 t_1, t_2 拼接到链接中，可以通过子任务 1。

全球知名算法竞赛：子任务 $1 \sim 2$

从第三个字符开始扫描字符串，如果是字母就加入字符串 t_1，否则加入字符串 t_2。最后把 t_1, t_2 拼接到链接中，可以通过子任务 1。

从第三个字符开始扫描字符串，把字符依次加入字符串 t_2，遇到字母之后的所有字符都加入 t_1，可以通过此题。

找倍数

给定一个序列 $a_1, a_2, \cdots, a_n$，请求出满足下述条件的所有下标 i $(1 \leq i \leq n)$：

- 不存在下标 $1 \leq u < v \leq n$，使得 a_i 既不是 a_u 的倍数，也不是 a_v 的倍数。

$1 \leq n \leq 2 \times 10^5$，$1 \leq a_i \leq 10^9$。

找倍数：子任务 $1 \sim 3$

子任务 1：$1 \leq n \leq 300$。

找倍数：子任务 $1 \sim 3$

子任务 1： $1 \leq n \leq 300$。
枚举 i，枚举 u, v 判断即可。时间复杂度 $O(n^3)$。

找倍数：子任务 $1 \sim 3$

子任务 1： $1 \leq n \leq 300$。
枚举 i，枚举 u, v 判断即可。时间复杂度 $O(n^3)$。
子任务 2： $1 \leq n \leq 2 \times 10^3$。

找倍数：子任务 $1 \sim 3$

子任务 1：$1 \leq n \leq 300$。
枚举 i，枚举 u, v 判断即可。时间复杂度 $O(n^3)$。
子任务 2：$1 \leq n \leq 2 \times 10^3$。
容易发现这个条件等价于至多存在一个不是 a_i 的因数的数。枚举 i，枚举另一个数判断即可。时间复杂度 $O(n^2)$。

找倍数：子任务 $1 \sim 3$

子任务 1：$1 \leq n \leq 300$。
枚举 i，枚举 u, v 判断即可。时间复杂度 $O(n^3)$。
子任务 2：$1 \leq n \leq 2 \times 10^3$。
容易发现这个条件等价于至多存在一个不是 a_i 的因数的数。枚举 i，枚举另一个数判断即可。时间复杂度 $O(n^2)$。
子任务 3：$1 \leq a_i \leq 10^6$。

找倍数：子任务 $1 \sim 3$

子任务 1：$1 \leq n \leq 300$。
枚举 i，枚举 u, v 判断即可。时间复杂度 $O(n^3)$。
子任务 2：$1 \leq n \leq 2 \times 10^3$。
容易发现这个条件等价于至多存在一个不是 a_i 的因数的数。枚举 i，枚举另一个数判断即可。时间复杂度 $O(n^2)$。
子任务 3：$1 \leq a_i \leq 10^6$。
用 10^6 个 `std::vector` 存储不同值在数列中的下标，枚举数 i 和其倍数 j，将值为 i 的数的个数加上 j，就可以求出每个数在数列中的因数个数。判断因数个数是否至少为 $n-1$ 即可，时间复杂度 $O(n \log n + v \log v)$。

找倍数：子任务 $4 \sim 5$

子任务 4：保证 a_i 互不相同。

找倍数：子任务 $4 \sim 5$

子任务 4：保证 a_i 互不相同。
注意到一个数是很多数的倍数，那么它在序列中一定是比较大的。显然符合条件的数只可能是是最大的两个数值，暴力判断即可。时间复杂度 $\Theta(n)$。

找倍数：子任务 $4 \sim 5$

子任务 4：保证 a_i 互不相同。
注意到一个数是很多数的倍数，那么它在序列中一定是比较大的。显然符合条件的数只可能是是最大的两个数值，暴力判断即可。时间复杂度 $\Theta(n)$。
在子任务 2 的基础上，如果已经枚举到至少两个数不是 a_i 的因数时使用 `break` 退出循环。由于保证数互不相同，所以至多有大约 10^3 个数是一个数的因数，似乎很难卡满。

找倍数：子任务 $4 \sim 5$

子任务 4：保证 a_i 互不相同。
注意到一个数是很多数的倍数，那么它在序列中一定是比较大的。显然符合条件的数只可能是是最大的两个数值，暴力判断即可。时间复杂度 $\Theta(n)$。
在子任务 2 的基础上，如果已经枚举到至少两个数不是 a_i 的因数时使用 `break` 退出循环。由于保证数互不相同，所以至多有大约 10^3 个数是一个数的因数，似乎很难卡满。
子任务 5：无特殊限制。

找倍数：子任务 $4 \sim 5$

子任务 4：保证 a_i 互不相同。
注意到一个数是很多数的倍数，那么它在序列中一定是比较大的。显然符合条件的数只可能是是最大的两个数值，暴力判断即可。时间复杂度 $\Theta(n)$。
在子任务 2 的基础上，如果已经枚举到至少两个数不是 a_i 的因数时使用 `break` 退出循环。由于保证数互不相同，所以至多有大约 10^3 个数是一个数的因数，似乎很难卡满。
子任务 5：无特殊限制。
与子任务 4 不同的是，你要判断所有与最大的两个数相等的位置。

5M 星球

构造一个 n 元集合的所有大小为奇数的子集到所有大小为偶数的子集的双射。

m 次询问每个子集所对应的子集。$1 \leq m \leq 5 \times 10^4$，$1 \leq n \leq 50$。

5M 星球：题解

子任务 1：$n \leq 20$。

5M 星球：题解

子任务 1：$n \leq 20$。
把所有子集列出来，任意配对即可。

5M 星球：题解

子任务 1：$n \leq 20$。
把所有子集列出来，任意配对即可。
子任务 3：保证 n 是奇数。

5M 星球：题解

子任务 1：$n \leq 20$。
把所有子集列出来，任意配对即可。
子任务 3：保证 n 是奇数。
把输入的 0 全部变为 1，1 全部变为 0 即可。

5M 星球：题解

子任务 1：$n \leq 20$。
把所有子集列出来，任意配对即可。
子任务 3：保证 n 是奇数。
把输入的 0 全部变为 1，1 全部变为 0 即可。
子任务 4：无特殊限制。

5M 星球：题解

子任务 1：$n \leq 20$。
把所有子集列出来，任意配对即可。
子任务 3：保证 n 是奇数。
把输入的 0 全部变为 1，1 全部变为 0 即可。
子任务 4：无特殊限制。
钦定一个原集合的元素 x，如果给定的子集没有 x 就加上 x，否则删掉 x。这也证明了大小为奇数的子集和大小为偶数的子集数量相等。

5M 星球：题解

子任务 1：$n \leq 20$。
把所有子集列出来，任意配对即可。
子任务 3：保证 n 是奇数。
把输入的 0 全部变为 1，1 全部变为 0 即可。
子任务 4：无特殊限制。
钦定一个原集合的元素 x，如果给定的子集没有 x 就加上 x，否则删掉 x。这也证明了大小为奇数的子集和大小为偶数的子集数量相等。
本题在赛时存在若干类似的做法，下面介绍一个赛时某选手在最后 1 分钟通过此题所使用的做法。

5M 星球：题解

不妨令第 k 小的 1 的数量为奇数的数对应第 k 小的 1 的数量为偶数的数。

5M 星球：题解

不妨令第 k 小的 1 的数量为奇数的数对应第 k 小的 1 的数量为偶数的数。
考虑数位 dp。先求出 $f(i,j)$ 表示前 i 个字符中 1 的数量模 2 同余于 j 的方案数。

5M 星球：题解

不妨令第 k 小的 1 的数量为奇数的数对应第 k 小的 1 的数量为偶数的数。
考虑数位 dp。先求出 $f(i,j)$ 表示前 i 个字符中 1 的数量模 2 同余于 j 的方案数。
然后处理一个数只需要做一遍 dp，依次确定每个位置的答案即可。

小老虎训练中心

第一次运行给定一个长度为 n 的字符串 s 和一个长度为 m 的随机生成的数列 a，你需要确定一个 a 的子序列 a'。然后评测系统对于每个子序列中的数都有 $\frac{1}{2}$ 的概率被删除。第二次运行给定删除后的序列，你需要还原字符串 s。

保证 $1 \leq n \leq 40$，$m = 3 \times 10^4$，a_i 从 $[1, 10^9]$ 中均匀随机。

小老虎训练中心：子任务 $1, 4$

子任务 1：保证 $n = 1$。

小老虎训练中心：子任务 $1, 4$

子任务 1：保证 $n = 1$。
容易想到用模 26 的余数来表示是什么字符，从给定数列中找出所有表示这个字符的数即可。期望长度 $\frac{3 \times 10^4}{26}$，每个数只有 $\frac{1}{2}$ 的概率被删除，可以通过。

小老虎训练中心：子任务 $1, 4$

子任务 1：保证 $n = 1$。
容易想到用模 26 的余数来表示是什么字符，从给定数列中找出所有表示这个字符的数即可。期望长度 $\frac{3 \times 10^4}{26}$，每个数只有 $\frac{1}{2}$ 的概率被删除，可以通过。
子任务 4：保证给定的字符串中，任意相邻的两个字符不相同。

小老虎训练中心：子任务 $1, 4$

子任务 1：保证 $n = 1$。
容易想到用模 26 的余数来表示是什么字符，从给定数列中找出所有表示这个字符的数即可。期望长度 $\frac{3 \times 10^4}{26}$，每个数只有 $\frac{1}{2}$ 的概率被删除，可以通过。
子任务 4：保证给定的字符串中，任意相邻的两个字符不相同。
考虑把给定序列分为 n 段，第 i 段找出所有表示这个字符的数即可，期望能找到 $\frac{3 \times 10^4}{26 \times 40}$ 约等于 29 个，可以通过。

小老虎训练中心：子任务 5

子任务 5：无特殊限制。

小老虎训练中心：子任务 5

子任务 5：无特殊限制。
子任务 4 的做法在相邻字符相同时可能出现问题，一个特殊字符 . 表示和上一个字符相同，这样可以保证相邻字符不相同，字符集大小变大 1，仍然可以通过。

小老虎训练中心：子任务 5 其他做法

子任务 5：无特殊限制。

DeaphetS[1] 的做法：若第 i 个字符是 s_i（此处 $0 \le s_i < 26$），则设 S 是所有 $26 \times i + s_i$ 组成的集合。从给定序列中找出所有模 40×26 的余数在 S 中的数即可，期望能找到 $\frac{3 \times 10^4}{26 \times 40}$ 约等于 29 个，可以通过。

[1] 在洛谷关注 DeaphetS 可以获得回关！

小老虎训练中心：子任务 5 其他做法

子任务 5：无特殊限制。
DeaphetS[1] 的做法：若第 i 个字符是 s_i（此处 $0 \le s_i < 26$），则设 S 是所有 $26 \times i + s_i$ 组成的集合。从给定序列中找出所有模 40×26 的余数在 S 中的数即可，期望能找到 $\frac{3 \times 10^4}{26 \times 40}$ 约等于 29 个，可以通过。
大众做法是每个字符后面都插入一个特殊字符来识别，因为限制比较松，所以可以通过。

[1] 在洛谷关注 DeaphetS 可以获得回关！

5M 宇宙

给定一个长度为 n 的包含问号 ？ 或小写字母的字符串 s。你需要把所有的 ？ 替换成小写字母，使得字符串中正好存在 m 个子序列 `wmy`。

5M 宇宙：子任务 $1 \sim 4$

子任务 1：保证 $1 \leq n \leq 4$。

5M 宇宙：子任务 $1 \sim 4$

子任务 1：保证 $1 \leq n \leq 4$。
枚举所有情况即可，时间复杂度 $O(26^n n^3)$。

5M 宇宙：子任务 $1 \sim 4$

子任务 1：保证 $1 \leq n \leq 4$。
枚举所有情况即可，时间复杂度 $O(26^n n^3)$。
子任务 2：保证字符串 s 不包含字符 ？。

5M 宇宙：子任务 $1 \sim 4$

子任务 1：保证 $1 \leq n \leq 4$。
枚举所有情况即可，时间复杂度 $O(26^n n^3)$。
子任务 2：保证字符串 s 不包含字符 ？。
相当于计数，枚举三个位置即可，时间复杂度 $O(n^3)$。

5M 宇宙：子任务 $1 \sim 4$

子任务 1：保证 $1 \leq n \leq 4$。
枚举所有情况即可，时间复杂度 $O(26^n n^3)$。
子任务 2：保证字符串 s 不包含字符 ？。
相当于计数，枚举三个位置即可，时间复杂度 $O(n^3)$。
子任务 3：保证 $1 \leq n \leq 8$。

5M 宇宙：子任务 $1 \sim 4$

子任务 1：保证 $1 \leq n \leq 4$。
枚举所有情况即可，时间复杂度 $O(26^n n^3)$。
子任务 2：保证字符串 s 不包含字符 ？。
相当于计数，枚举三个位置即可，时间复杂度 $O(n^3)$。
子任务 3：保证 $1 \leq n \leq 8$。
因为如果一个 ？ 不是 w、m、y 三者时等价，所以每个位置只有 4 种情况，枚举即可。时间复杂度 $O(4^n n^3)$。

5M 宇宙：子任务 $1 \sim 4$

子任务 1：保证 $1 \leq n \leq 4$。
枚举所有情况即可，时间复杂度 $O(26^n n^3)$。
子任务 2：保证字符串 s 不包含字符 ？。
相当于计数，枚举三个位置即可，时间复杂度 $O(n^3)$。
子任务 3：保证 $1 \leq n \leq 8$。
因为如果一个 ？ 不是 w、m、y 三者时等价，所以每个位置只有 4 种情况，枚举即可。时间复杂度 $O(4^n n^3)$。
子任务 4：保证 $1 \leq n \leq 28$。

5M 宇宙：子任务 $1 \sim 4$

子任务 1：保证 $1 \leq n \leq 4$。
枚举所有情况即可，时间复杂度 $O(26^n n^3)$。
子任务 2：保证字符串 s 不包含字符 ？。
相当于计数，枚举三个位置即可，时间复杂度 $O(n^3)$。
子任务 3：保证 $1 \leq n \leq 8$。
因为如果一个 ？ 不是 w、m、y 三者时等价，所以每个位置只有 4 种情况，枚举即可。时间复杂度 $O(4^n n^3)$。
子任务 4：保证 $1 \leq n \leq 28$。
如果后面的子任务的做法实现不够优秀，仍然可以通过该子任务。

5M 宇宙：子任务 5

子任务 5：$n \leq 40$，$m \leq 80$。

5M 宇宙：子任务 5

子任务 5：$n \leq 40$，$m \leq 80$。
考虑动态规划。

5M 宇宙：子任务 5

子任务 5：$n \leq 40$，$m \leq 80$。
考虑动态规划。
设 $f(i, W, M, Y)$ 表示前 i 个字符中，有 W 个字符 `w`，有 M 个子序列 `wm`，有 Y 个子序列 `wmy` 是否可行。

5M 宇宙：子任务 5

子任务 5：$n \leq 40$，$m \leq 80$。
考虑动态规划。
设 $f(i, W, M, Y)$ 表示前 i 个字符中，有 W 个字符 `w`，有 M 个子序列 `wm`，有 Y 个子序列 `wmy` 是否可行。
构造只需要倒推即可。

5M 宇宙：子任务 5

子任务 5：$n \leq 40$，$m \leq 80$。
考虑动态规划。
设 $f(i, W, M, Y)$ 表示前 i 个字符中，有 W 个字符 `w`，有 M 个子序列 `wm`，有 Y 个子序列 `wmy` 是否可行。
构造只需要倒推即可。
注意到只有 $W, M, Y \leq m$ 时状态有效，时间复杂度 $\Theta(nm^3)$。

5M 宇宙：子任务 6

子任务 6：$n \leq 40$。

5M 宇宙：子任务 6

子任务 6：$n \leq 40$。
可以把子序列的贡献算在字符 m 上，贡献为之前的 w 的数量乘之后的 y 的数量。

5M 宇宙：子任务 6

子任务 6：$n \leq 40$。
可以把子序列的贡献算在字符 m 上，贡献为之前的 w 的数量乘之后的 y 的数量。
仍然考虑动态规划，设 $f(i, j, W, Y)$ 表示前 i 个字符中的 m 贡献的子序列数为 j，W 表示前 i 个字符有多少个 w，Y 表示后 $n - i$ 个字符有多少个 y 是否可行。转移时枚举当前位置所填的字符，构造只需要倒推即可。时间复杂度 $\Theta(n^3 m)$。

5M 宇宙：子任务 6

子任务 6：$n \leq 40$。
可以把子序列的贡献算在字符 m 上，贡献为之前的 w 的数量乘之后的 y 的数量。
仍然考虑动态规划，设 $f(i, j, W, Y)$ 表示前 i 个字符中的 m 贡献的子序列数为 j，W 表示前 i 个字符有多少个 w，Y 表示后 $n - i$ 个字符有多少个 y 是否可行。转移时枚举当前位置所填的字符，构造只需要倒推即可。时间复杂度 $\Theta(n^3 m)$。
注意到子任务 5 的解法中状态表示的只是是否可行，也可以使用 `std::bitset` 优化来通过子任务 6，时间复杂度 $\Theta\left(\frac{nm^3}{w}\right)$。

5M 宇宙：子任务 8

子任务 8：无特殊限制。

5M 宇宙：子任务 8

子任务 8：无特殊限制。

注意到子任务 6 中状态表示的只是是否可行，可以使用 `std::bitset` 优化，时间复杂度 $\Theta\left(\frac{n^3m}{w}\right)$。

念经 II

给定 n, x, y，构造两个长度为 n、只包含 0 或者 1 的、分别包含 x, y 个 1 的字符串 a, b，最小化在 b 任意循环位移时，a, b 均为 1 的位置的最大值。

$1 \leq n \leq 5 \times 10^5$，$0 \leq x, y \leq n$。

念经 II：子任务 $1 \sim 2$

子任务 1： $n \leq 12$。

念经 II：子任务 $1 \sim 2$

子任务 1：$n \leq 12$。
枚举所有情况即可。

念经 II：子任务 $1 \sim 2$

子任务 1： $n \leq 12$。
枚举所有情况即可。
子任务 2： $x, y \leq 5$，$n \geq 100$。

念经 II：子任务 $1 \sim 2$

子任务 1：$n \leq 12$。
枚举所有情况即可。
子任务 2：$x, y \leq 5$，$n \geq 100$。
在 $x \times y \leq n$ 时，第一行显然为 1，因为可以构造 a 中包含连续 x 个 1，b 中每 x 个位置放一个 1。

念经 II：求解第一行

注意到每一对 a 中的 1 和 b 中的 1 都正好在一次循环移位中被匹配。

念经 II：求解第一行

注意到每一对 a 中的 1 和 b 中的 1 都正好在一次循环移位中被匹配。故 n 次循环移位中共匹配了 xy 次，由抽屉原理知最优情况下答案为 $\left\lceil \frac{xy}{n} \right\rceil$。

念经 II：求解第一行

注意到每一对 a 中的 1 和 b 中的 1 都正好在一次循环移位中被匹配。故 n 次循环移位中共匹配了 xy 次，由抽屉原理知最优情况下答案为 $\left\lceil \frac{xy}{n} \right\rceil$。当然，你也可以打表找出这个答案。

念经 II：子任务 $4 \sim 5$

子任务 4：$\gcd(n, x) = 1$。

念经 II：子任务 $4 \sim 5$

子任务 4：$\gcd(n, x) = 1$。
不妨设 $x < y$。经过不断打表可以发现，一定存在一种构造使得 a 中 x 个 1 是连续的。当 $\gcd(n, x) = 1$ 时，只需在 b 中每 x 个位置放一个 1，到结尾了回到开头即可。

念经 II：子任务 $4 \sim 5$

子任务 4：$\gcd(n, x) = 1$。
不妨设 $x < y$。经过不断打表可以发现，一定存在一种构造使得 a 中 x 个 1 是连续的。当 $\gcd(n, x) = 1$ 时，只需在 b 中每 x 个位置放一个 1，到结尾了回到开头即可。
子任务 5：$n \leq 5 \times 10^3$。

念经 II：子任务 $4 \sim 5$

子任务 4：$\gcd(n, x) = 1$。
不妨设 $x < y$。经过不断打表可以发现，一定存在一种构造使得 a 中 x 个 1 是连续的。当 $\gcd(n, x) = 1$ 时，只需在 b 中每 x 个位置放一个 1，到结尾了回到开头即可。
子任务 5：$n \leq 5 \times 10^3$。
考虑在子任务 4 做法的基础上，如果下一次放 1 的位置被占用，就向后找到第一个没有放 1 的位置。

念经 II：求解第二行

解法一：把 b 按照长度为 $\gcd(n,x)$ 分段，记录每段 1 的数量，同样按照子任务 4 的做法，每 $\dfrac{x}{\gcd(n,x)}$ 段将这个段 $+1$ 即可。

念经 II：求解第二行

解法一：把 b 按照长度为 $\gcd(n,x)$ 分段，记录每段 1 的数量，同样按照子任务 4 的做法，每 $\dfrac{x}{\gcd(n,x)}$ 段将这个段 $+1$ 即可。

解法二：考虑限制前缀和。

念经 II：求解第二行

解法一：把 b 按照长度为 $\gcd(n,x)$ 分段，记录每段 1 的数量，同样按照子任务 4 的做法，每 $\dfrac{x}{\gcd(n,x)}$ 段将这个段 $+1$ 即可。

解法二：考虑限制前缀和。只需令前 i 个数之和为 $\min\left(\left\lfloor\dfrac{A\times i}{x}\right\rfloor,y\right)$，其中 A 表示答案，也就是连续 x 个字符中 1 的数量的上限。

垃圾桶

有 $2n$ 个数分为两组，每组 n 个随机生成，保证总和为 m。所有数随机打乱之后你需要找出任意一个集合总和为 m。
$1 \leq n \leq 50$，存在 $1 \leq k \leq 12$ 使得 $m = 10^k$。

垃圾桶：算法 $1 \sim 3$

算法 1

暴力枚举所有子集。时间复杂度 $O(2^{2n}n)$，预期可以通过 $n \leq 9$ 的情况。

垃圾桶：算法 $1 \sim 3$

算法 1

暴力枚举所有子集。时间复杂度 $O(2^{2n}n)$，预期可以通过 $n \leq 9$ 的情况。

算法 2

考虑使用 Meet in the Middle，把序列分为两半分别枚举所有子集。时间复杂度 $O(2^n n)$，预期可以通过 $n \leq 18$ 的情况。

垃圾桶：算法 $1 \sim 3$

算法 1

暴力枚举所有子集。时间复杂度 $O(2^{2n}n)$，预期可以通过 $n \leq 9$ 的情况。

算法 2

考虑使用 Meet in the Middle，把序列分为两半分别枚举所有子集。时间复杂度 $O(2^n n)$，预期可以通过 $n \leq 18$ 的情况。

算法 3

背包求解，构造只需要从后往前推即可。时间复杂度 $O(nm)$。预期可以通过 $m \leq 10^6$ 的情况。

垃圾桶：算法 $4 \sim 5$

算法 4

考虑优化算法 3，注意到求解背包时每个状态的值是布尔类型的，可以使用 `std::bitset` 优化，时间复杂度 $O\left(\frac{nm}{w}\right)$，预期可以通过 $m \leq 10^7$ 的情况。

垃圾桶：算法 $4 \sim 5$

算法 4

考虑优化算法 3，注意到求解背包时每个状态的值是布尔类型的，可以使用 `std::bitset` 优化，时间复杂度 $O\left(\frac{nm}{w}\right)$，预期可以通过 $m \leq 10^7$ 的情况。

算法 5

注意到深度优先搜索的时间复杂度是 $O(2^{2n})$ 的，考虑搜索的剪枝优化，预处理原数列的后缀和，从前向后做深度优先搜索，如果搜索过程中发现后缀和小于当前所分的两组数的差的绝对值就退出。预期可以通过 $n \leq 9$ 或者 $m \leq 10^9$ 的情况。

垃圾桶：算法 6

算法 6

考虑优化算法 2，注意到 Meet in the Middle 只能处理 36 个数的问题，可以把数列任意划分为 36 部分做 Meet in the Middle，这样有 2^{36} 种方案，远大于 10^9，预期可以通过 $n \leq 18$ 或者 $m \leq 10^9$ 的情况。

垃圾桶：算法 7

算法 7

任取一个或者两个素数，以它为模数做背包，构造方案时从后往前倒推。但是这样会遇到两种转移都可行的方案，任意选一种可能导致总和不为 m。由于数据随机，所以每次随机选择一个转移，反复随机直到找到方案即可。预期 [2] 可以通过 $n \leq 9$ 或者 $m \leq 10^{10}$ 的情况。

[2] 预测指的是稳定能够通过的情况，下同。实际评测时评测系统（Hydro）会在一个测试点 TLE 之后自动重测该测试点若干次，所以部分随机化算法在实际评测时效果比上述预测更加优秀。

垃圾桶：算法 $8 \sim 9$

算法 8

把序列分为两半，不断从两部分分别随机子集，用 `std::unordered_map` 存储已经找到过的总和和对应的元素编号，找到和为 m 的就输出。预期可以通过 $n \leq 18$ 或者 $m \leq 10^{11}$ 的情况。

垃圾桶：算法 $8 \sim 9$

算法 8

把序列分为两半，不断从两部分分别随机子集，用 `std::unordered_map` 存储已经找到过的总和和对应的元素编号，找到和为 m 的就输出。预期可以通过 $n \leq 18$ 或者 $m \leq 10^{11}$ 的情况。

算法 9

把序列分为两半，先在第一部分中随机 10^5 个子集，仍然用 `std::unordered_map` 存储，然后再不断随机第二部分的子集直到找到和为 m 的。预期可以通过 $n \leq 9$ 或者 $m \leq 10^{11}$ 的情况。

垃圾桶：算法 $10 \sim 11$

算法 10

不知道为什么，把算法 7 中的素数改为模 10^6 会更优，预期可以通过 $n \leq 18$ 或者 $m \leq 10^{11}$ 的情况。

垃圾桶：算法 $10 \sim 11$

算法 10

不知道为什么，把算法 7 中的素数改为模 10^6 会更优，预期可以通过 $n \leq 18$ 或者 $m \leq 10^{11}$ 的情况。

算法 11

考虑优化算法 10，类似地，背包时可以使用 `std::bitset` 优化，从而模数可以改为 10^7，提高正确率。可以通过此题。

垃圾桶：算法 12

算法 12

这是一个来自于赛时某选手的做法。
前若干个数随机选取，剩下的数 Meet in the Middle，卡一卡可以通过此题。

数学高手 II

设 m 为正整数，数列 $a_1, a_2, \cdots, a_{4m+2}$ 是公差不为 0 的等差数列，若从中删去两项 a_i 和 a_j $(i < j)$ 后剩余的 $4m$ 项可被平均分为 m 组，每组的 4 个数都能构成等差数列，则称数列 $a_1, a_2, \cdots, a_{4m+2}$ 是关于 (i, j) 的可分数列。

判断是否可划分并构造方案 $(1 \leq n \leq 10^5)$，或者求可划分的方案数量 $(1 \leq n \leq 10^9)$。

数学高手 II：题解

注意到要求拆成长度为 4 的等差数列，于是很难不想到把取出来的 4 个数的位置模 4 分析。

数学高手 II：题解

注意到要求拆成长度为 4 的等差数列，于是很难不想到把取出来的 4 个数的位置模 4 分析。

- 如果公差为偶数，那么模 4 余 $1,3$ 的位置数量奇偶性不变。

数学高手 II：题解

注意到要求拆成长度为 4 的等差数列，于是很难不想到把取出来的 4 个数的位置模 4 分析。

- 如果公差为偶数，那么模 4 余 $1,3$ 的位置数量奇偶性不变。
- 如果公差为奇数，那么模 4 余 $1,3$ 的位置数量奇偶性均变化。

数学高手 II：题解

注意到要求拆成长度为 4 的等差数列，于是很难不想到把取出来的 4 个数的位置模 4 分析。

- 如果公差为偶数，那么模 4 余 $1,3$ 的位置数量奇偶性不变。
- 如果公差为奇数，那么模 4 余 $1,3$ 的位置数量奇偶性均变化。

由此可以得出模 4 余 $1,3$ 的位置数量奇偶性相同，同理，余 $0,2$ 的位置数量奇偶性也相同。

数学高手 II：题解

注意到要求拆成长度为 4 的等差数列，于是很难不想到把取出来的 4 个数的位置模 4 分析。

- 如果公差为偶数，那么模 4 余 $1,3$ 的位置数量奇偶性不变。
- 如果公差为奇数，那么模 4 余 $1,3$ 的位置数量奇偶性均变化。

由此可以得出模 4 余 $1,3$ 的位置数量奇偶性相同，同理，余 $0,2$ 的位置数量奇偶性也相同。
注意到 $4m+2$ 个位置中，模 4 余 $1,2$ 的位置多一个，所以可以得出结论：可划分的必要条件是删去的两个位置模 4 余 $1,2$。

数学高手 II：题解

当靠前的位置模 4 余 1 时，显然可以令三段都分为连续的四个位置。

数学高手 II：题解

当靠前的位置模 4 余 1 时，显然可以令三段都分为连续的四个位置。
当靠前的位置模 4 余 2 时，考虑在第一段和最后一段分别取出尽可能多的连续四个数为一组。问题转化为一个长度为 $4m+2$ 的序列，删去 2 和 $4m+1$ 后如何划分。

数学高手 II：题解

当靠前的位置模 4 余 1 时，显然可以令三段都分为连续的四个位置。
当靠前的位置模 4 余 2 时，考虑在第一段和最后一段分别取出尽可能多的连续四个数为一组。问题转化为一个长度为 $4m+2$ 的序列，删去 2 和 $4m+1$ 后如何划分。
考虑划分为 m 个公差为 m 的数列，位置 $1, 2, \cdots, m$ 分别为 m 个数列的开头。这样就把 $1 \sim 4m$ 划分为 m 个公差为 m 的数列。

数学高手 II：题解

当靠前的位置模 4 余 1 时，显然可以令三段都分为连续的四个位置。
当靠前的位置模 4 余 2 时，考虑在第一段和最后一段分别取出尽可能多的连续四个数为一组。问题转化为一个长度为 $4m+2$ 的序列，删去 2 和 $4m+1$ 后如何划分。
考虑划分为 m 个公差为 m 的数列，位置 $1, 2, \cdots, m$ 分别为 m 个数列的开头。这样就把 $1 \sim 4m$ 划分为 m 个公差为 m 的数列。但是现在要删掉 2，所以在包含 2 的数列中删掉 2，加上 $4m+2$ 即可正好划分完。

数学高手 II：题解

但是注意到存在一种特殊情况：两个删除的位置的差为 3，此时无法使用上述方法划分。

数学高手 II：题解

但是注意到存在一种特殊情况：两个删除的位置的差为 3，此时无法使用上述方法划分。
经过枚举可以发现：

- $m = 6$ 时，删去的位置为 $10, 13$ 或者 $14, 17$ 时存在一组解。

数学高手 II：题解

但是注意到存在一种特殊情况：两个删除的位置的差为 3，此时无法使用上述方法划分。
经过枚举可以发现：

- $m = 6$ 时，删去的位置为 $10, 13$ 或者 $14, 17$ 时存在一组解。也就是说，这两个删除的位置两侧分别多出来至少 9 和 13 个位置时存在解。

数学高手 II：题解

但是注意到存在一种特殊情况：两个删除的位置的差为 3，此时无法使用上述方法划分。
经过枚举可以发现：

- $m = 6$ 时，删去的位置为 $10, 13$ 或者 $14, 17$ 时存在一组解。也就是说，这两个删除的位置两侧分别多出来至少 9 和 13 个位置时存在解。
- $m = 7$ 时，删去的位置为 $6, 9$ 时存在一组解。也就是说，这两个删除的位置两侧分别多出来至少 5 和 21 个位置时存在解。

数学高手 II：题解

但是注意到存在一种特殊情况：两个删除的位置的差为 3，此时无法使用上述方法划分。
经过枚举可以发现：

- $m = 6$ 时，删去的位置为 $10, 13$ 或者 $14, 17$ 时存在一组解。也就是说，这两个删除的位置两侧分别多出来至少 9 和 13 个位置时存在解。
- $m = 7$ 时，删去的位置为 $6, 9$ 时存在一组解。也就是说，这两个删除的位置两侧分别多出来至少 5 和 21 个位置时存在解。
- $m = 8$ 时，删去的位置为 $2, 5$ 时存在一组解。也就是说，这两个删除的位置两侧分别多出来至少 1 和 29 个位置时存在解。

数学高手 II：题解

但是注意到存在一种特殊情况：两个删除的位置的差为 3，此时无法使用上述方法划分。
经过枚举可以发现：

- $m = 6$ 时，删去的位置为 $10, 13$ 或者 $14, 17$ 时存在一组解。也就是说，这两个删除的位置两侧分别多出来至少 9 和 13 个位置时存在解。
- $m = 7$ 时，删去的位置为 $6, 9$ 时存在一组解。也就是说，这两个删除的位置两侧分别多出来至少 5 和 21 个位置时存在解。
- $m = 8$ 时，删去的位置为 $2, 5$ 时存在一组解。也就是说，这两个删除的位置两侧分别多出来至少 1 和 29 个位置时存在解。

由此不难得出在 $n \geq 8$ 时删除位置差为 3 的都存在解。

一些有趣的事情

- B 题的验题（口胡）阶段，验题人 jiazhichen844 和审核人 DeaphetS 均被硬控 10 分钟左右。这导致出现了你现在看到的 A 题。

[3] https://www.luogu.com/article/d3dysfnf

一些有趣的事情

- B 题的验题（口胡）阶段，验题人 jiazhichen844 和审核人 DeaphetS 均被硬控 10 分钟左右。这导致出现了你现在看到的 A 题。
- B 题的造题阶段，为了让 jiazhichen844 干活，某人提出了 Code Golf，比出题人和验题人的代码谁短。

[3] https://www.luogu.com/article/d3dysfnf

一些有趣的事情

- B 题的验题（口胡）阶段，验题人 jiazhichen844 和审核人 DeaphetS 均被硬控 10 分钟左右。这导致出现了你现在看到的 A 题。
- B 题的造题阶段，为了让 jiazhichen844 干活，某人提出了 Code Golf，比出题人和验题人的代码谁短。
- 如果你想了解更多的 C 和 E 题的题目背景，欢迎阅读 fangzichang 的文章 [3]。

[3] https://www.luogu.com/article/d3dysfnf

一些有趣的事情

- B 题的验题（口胡）阶段，验题人 jiazhichen844 和审核人 DeaphetS 均被硬控 10 分钟左右。这导致出现了你现在看到的 A 题。
- B 题的造题阶段，为了让 jiazhichen844 干活，某人提出了 Code Golf，比出题人和验题人的代码谁短。
- 如果你想了解更多的 C 和 E 题的题目背景，欢迎阅读 fangzichang 的文章 [3]。
- E 题原题中由于搬题人取得了赛时 + 赛后的最优解，所以一开始没考虑其他做法，后来 konata 提出了某做法，经过对于原题的研究，我们发现本题存在很多更高复杂度的做法通过，所以加大了数据范围并且缩小了时间限制。

[3] https://www.luogu.com/article/d3dysfnf

一些有趣的事情

- 批话哥 Umbrella Leaf 假装自己一开始不会 E 题。

一些有趣的事情

- 批话哥 Umbrella Leaf 假装自己一开始不会 E 题。
- 非常抱歉，A 题测试数据并不是故意有前导零，而是数据生成器中翻转字符串使用了 `str.reverse()`，导致字符串并没有被翻转。

一些有趣的事情

- 批话哥 Umbrella Leaf 假装自己一开始不会 E 题。
- 非常抱歉，A 题测试数据并不是故意有前导零，而是数据生成器中翻转字符串使用了 `str.reverse()`，导致字符串并没有被翻转。
- 验题人 jiazhichen844 拿着题解差点过不去 G 题。

一些有趣的事情

- 批话哥 Umbrella Leaf 假装自己一开始不会 E 题。
- 非常抱歉，A 题测试数据并不是故意有前导零，而是数据生成器中翻转字符串使用了 `str.reverse()`，导致字符串并没有被翻转。
- 验题人 jiazhichen844 拿着题解差点过不去 G 题。
- 本场比赛预期结果是有 $2 \sim 3$ 个选手能够通过 $6 \sim 8$ 题。

一些有趣的事情

- 批话哥 Umbrella Leaf 假装自己一开始不会 E 题。
- 非常抱歉，A 题测试数据并不是故意有前导零，而是数据生成器中翻转字符串使用了 `str.reverse()`，导致字符串并没有被翻转。
- 验题人 jiazhichen844 拿着题解差点过不去 G 题。
- 本场比赛预期结果是有 $2 \sim 3$ 个选手能够通过 $6 \sim 8$ 题。
- H 题审核人 DeaphetS 验题时花费了整整 3 个小时左右的时间才取得 AC。

一些有趣的事情

- 批话哥 Umbrella Leaf 假装自己一开始不会 E 题。
- 非常抱歉，A 题测试数据并不是故意有前导零，而是数据生成器中翻转字符串使用了 `str.reverse()`，导致字符串并没有被翻转。
- 验题人 jiazhichen844 拿着题解差点过不去 G 题。
- 本场比赛预期结果是有 $2 \sim 3$ 个选手能够通过 $6 \sim 8$ 题。
- H 题审核人 DeaphetS 验题时花费了整整 3 个小时左右的时间才取得 AC。
- 本来本场比赛还有 I 题《基础贪心练习题 / 堆石头》，但是由于 I 题比较适合放在模拟赛，正好题目比较多，所以删减了。

吐槽

大家来吐槽一下吧！
结束之后将会在 GitHub 仓库 [4] 公布题目资源。
预告：

- 我和不知名用户 [5] 将在近期准备一场包含“猴子进化 II”的 NOIP 模拟赛。
- Zzzcr [6] 预计在获得 CSP-S 一等奖之后举办一场梦熊周赛未来组。
- 预计明年同期出一场全部原创的周赛。

[4] https://github.com/Molmin/contest-20240817
[5] 不知名用户的个人中心 - 洛谷
[6] Zzzcr 的个人中心 - 洛谷